International Scholarly Research Network
ISRN Signal Processing
Volume 2011, Article ID 975145, 17 pages
doi:10.5402/2011/975145

Research Article

TV Stream Structuring

Zein Al Abidin Ibrahim[1] and Patrick Gros[2]

[1] *LERIA Laboratory, Angers University, 49045 Angers, France*
[2] *INRIA, Centre Rennes-Bretagne Atlantique, 35042 Rennes, France*

Correspondence should be addressed to Patrick Gros, patrick.gros@inria.fr

Received 3 March 2011; Accepted 19 April 2011

Academic Editors: H. Araujo and W.-L. Hwang

TV stream structuring consists in detecting precisely the first and the last frames of all the programs and the breaks (commercials, trailers, station identification, bumpers) of a given stream and then in annotating all these segments with metadata. Usually, breaks are broadcasted several times during a stream. Thus, the detection of these repetitions can be considered as a key tool for stream structuring. After the detection stage, a classification method is applied to separate the repetitions in programs and breaks. In their turn, breaks repetitions are then used to classify the segments which appear only once in the stream. Finally, the stream is aligned with an electronic program guide (EPG), in order to annotate the programs. Our experiments have been applied on a 22-day long TV stream, and results show the efficiency of the proposed method in TV stream structuring.

1. Introduction

With the rapid development of digital technologies in the last decades, capturing and storing digital video content have become very common and easy. Television is probably the main source of such videos. However, navigating within TV streams is still a very complicated task to achieve. Users conceive a TV stream as a sequence of programs (P) and breaks (B), while from a signal point of view, this stream is seen as a continuous flow of frames and sounds, with no external markers of the start and end points of the included programs and no apparent structure. Most of TV streams have no associated metadata to describe their structure, except the program guide produced by TV channels. This program guide lacks precision, since the TV channels cannot predict the exact duration of live programs for example. In addition, it does not provide any information about breaks like commercials or trailers. The first step in the stream structuring chain is to segment the stream in TV programs by providing theirs precise boundaries. The second step consists in recognizing each program and in describing its actual content if needed.

In addition to program retrieval, TV stream structuring is very valuable for several applications. One example is archive management as it is done at the French National Audiovisual Institute (INA). This institute is responsible for archiving all programs for which some French money was invested, and, in order to achieve this goal, it records all French TV channels 24/7. Since September 2006, INA has archived 540,000 hours of TV per year [1]. This amount of data is increasing since the number of channels is in progress. Accessing the data is thus a crucial problem. To facilitate this task, INA manually describes the structure and the content of each stream.

Another application of TV stream structuring is the analysis of commercial breaks, either to enforce legal regulations, to get some statistics, or to skip these commercials.

Moreover, nowadays video analysis techniques may face a major problem. Mostly of these techniques are designed to analyze a video that contains homogeneous content (content of same type, i.e., soccer game), while TV streams may contain several heterogeneous programs. The TV stream structuring can be used by video analysis techniques in this case to segment the stream in programs and then applying the adequate technique for each program basing on its type. In addition to that, Internet Protocol Television (IPTV) services may use the TV stream structuring techniques in the catch-up TV, start-over TV, TVoD, or nPVR services.

Several methods have already been proposed to delimit or detect some specific content items in TV streams. Some detect bumpers to find breaks or programs. Others are

dedicated to commercials. Each of the existing methods solves only a part of the structuring problem. An example of the first complete solutions is Naturel's [1]. Nevertheless, this approach requires a lot of manual annotation and cannot scale up. On the other hand, Manson and Berrani [2] proposed a new technique based on a supervised learning algorithm, but this technique also requires manual annotation in order to train the system. Moreover, Manson and et al. classify each segment independently from its repetitions even if they probably have the same content type (P or B). Furthermore, Poli [3] proposed a top-down approach, which learns to predict a more accurate program guide. This approach in its turn requires an enormous learning set (several years of exact program guides in his case).

The complete and pioneer method proposed by Naturel et al. [1] is based on a manually annotated reference video database which is considered as its main limitation. Our approach is an extension of Naturel's method which tries to suppress the manual annotation stage of his method, and to reduce the manual annotation work of supervised methods. It starts by detecting the segments that appear several times in the stream. Then, a classification method separates the program segments (P) from break segments (B), which are used later to segment the complete stream into program and break segments. Finally, the EPG is used to label the program segments.

The rest of this paper is organized as follows. In Section 2, we present an overview of the existing TV structuring methods. To facilitate the comprehension of our method, a vocabulary is defined in Section 3. Section 4 details our proposed method. Experimentations and results are provided in Section 5. Finally, we conclude in Section 6.

2. State of the Art

The TV stream structuring problem has not been extensively addressed in the literature. Most of the previous works focus on structuring a single program or a collection of programs without dealing with streams containing several heterogeneous programs. However, the literature is rich with systems that are dedicated to detect commercials which could be considered as the basis of any TV stream structuring system (i.e., [4–9]). However, these techniques are not sufficient to structure the streams because commercials are not the only kind of breaks.

Returning to the TV stream structuring process, it can be divided into two complementary tasks.

(i) The first task consists in segmenting the stream in Program/Break sequences where the precise start and end of programs and breaks are provided.

(ii) In the second task, each segmented program is labeled with metadata in order to identify it and to facilitate the retrieval of information from the stream.

The first task of the process can be done based on different approaches.

(i) Segmenting the stream into logical units and then classifying each segment as a program or a break segment [2]. These segments may be of different granularities (Key-frame, Shot, Scene,…). Then, consecutive segments of the same content (same commercial, e.g.,) are combined.

(ii) Searching the start and the end of program segments basing on the detection of discontinuities in the homogeneities of some features [10], modeling the boundary between programs and breaks [11], or detecting the repetition of opening and closing credits [12].

(iii) Searching the start and the end of break segments by recognizing them in a reference database [1] or basing on their repeated behavior [2, 13]. The latter should be followed by a classification step in order to separate repeated program segments from break ones.

The existing approaches of the literature may be of two types.

(1) Metadata-based: Methods that are based on the almost exclusive use of the metadata available with the stream in order to structure the TV stream [3].

(2) Content-based: Methods that use the audiovisual stream to structure TV streams. In their turn, these methods can be classified into two subclasses.

(i) Methods that search the boundaries of the programs themselves. This type of methods is noted as program-based methods [10–12, 14]

(ii) Methods that search to detect breaks, which may separate consecutive programs. These methods are called break-based methods [1, 2, 13].

We classify the methods of the literature in four categories based on the techniques they rely on.

Category 1. A prototype of the first category is the metadata-based method developed by Poli [3] in order to structure TV streams. His main idea is to use a large set of already annotated data to learn a model of the program guide and thus of the stream structure. A hidden Markov model and a decision tree are used to learn this model that predicts the start time and the genre of all programs and breaks appearing during a week. This is the only method that is totally based on television schedules, and it requires a huge amount of annotated data for the learning stage (up to one year for each channel). An additional step may be required afterward to analyze the stream since the prediction is not perfect. This work proved that, on the channels used, the stream structure is very stable over the years.

Category 2. The second category contains the program-based methods that recover the structure of the TV stream by detecting the programs boundaries [10–12, 14]. In [12], the authors start from the assumption that, when considering two consecutive days, a given program starts approximately at the same time with the same opening and closing credits.

As a consequence, their method relies on the repetitive behavior of the open and closing credits of programs in order to detect their start and end time. The assumption used by the authors is not always true. Some programs do not have opening and closing credits. In addition, the TV channels broadcasts change completely in weekends. Likewise Liang et al. and Wang et al. propose in [11] a method basing on the opening and closing credits of programs. The idea is to detect special images called *Program-Oriented Informative iMages* (POIMs). These POIMs are frames containing logos with monochrome backgrounds and big text characters. From the authors' point of view, these POIMs appear in opening and closing credits and at the end of commercial segments. Unfortunately, opening and closing credits are not always present as mentioned before. Moreover, these POIM frames are not always present at the end of commercials and are variable from channel to channel. Contrarily to the methods proposed in [11, 12], El-Khoury et al. propose an original unsupervised method based on the fact that each program has homogeneous properties [10]. Consequently, the programs are extracted by detecting the discontinuities of some audiovisual features. The authors start from the idea that during a program, a selected set of features behaves in a homogeneous manner. In this method, short programs may not be detected and consecutive segments that belong to the same program are not merged. Moreover, detecting the boundaries of the breaks is easier and more precise than the program ones.

Category 3. In the third category, we find the recognition-based techniques that detect break segments. The work of Naturel et al. presented in [1] is the only complete structuring solution that is based on a reference database containing manually annotated breaks. This database is used to detect the breaks of the database rebroadcasted in the following part of the stream. The authors use hashing tables with video signatures in order to detect such repetitions which are used later to get the structure of the stream. The manual annotation of the database is the main constraint and drawback of the method. On the other hand, an automatic technique is proposed to update the database and thus to face the continuous change of the breaks. Unfortunately, the experimental data set used in this paper is not long enough to validate this updating approach.

Category 4. The techniques of the last category are the break-based methods that base on the detection of repeated audio-visual sequences in the TV stream. The underlying idea is that breaks and especially commercials have a repetitive behavior. Our technique falls into this category. However, several methods that use this principle have been already proposed. For example, Zeng et al. [13] use hashing tables with audio signatures in order to detect such repetitions. On the other hand, Berrani and Manson in [2, 16] use a clustering-based approach, which groups similar key-frames and visual features and then use inductive logic programming (ILP) to classify them into P and B segments. This last method, based on a supervised symbolic machine learning technique (ILP), shows that it is possible to learn the structure of the stream from the raw data. Thus, in somehow it makes a link between the Naturel's technique and the Poli's one. The drawback of this method is that it needs 7 days of manually annotated data to train the system. Moreover, ILP restricts the usable information to the local context of each segment. In addition to that, authors have chosen to classify each segment independently from its repetitions. From our point of view, most of the times, a segment and its repetitions are of the same type except in the trailers case. The trailers segments can be filtered using predefined rules. In our method, we take into account the contextual information of all occurrences in the stream of a given content. This last step adds a considerable improvement to the results as shown in the experimentations. Contrary to the methods in this category, the method proposed in [13] relays on audio signatures. Authors of [13] justify this choice by the fact that audio can overcome the limitation of the time-consuming video decoding. Using audio is a good idea, but it should be noticed that video decoding is not so time-consuming nowadays. Moreover, detecting audio segment boundaries is not quite easy. As a consequence, video signatures are used to overcome the latter problem. In addition, video signatures may be more robust than audio ones since the audio signal is very sensitive to noise and this may affect the repetition detection. On the other hand, the used video stream to evaluate the method is not long-enough, and the number of repetitions it contains is not provided. The rules used for the segmentation are very simple ones and their effectiveness is not clearly evaluated, for example, by a comparison with the ground truth. Finally, the programs segmented by this method have to be annotated manually. The metadata provided with the stream (EPG), which is an interesting source of information, is not used.

As a conclusion of this state-of-the-art survey, it should be noticed that the techniques based on the repetition detection are the more suitable ones as we search to segment the stream into programs and breaks. Of course, the breaks are not the only repetitive segments. Some program segments can appear several times in the stream, like opening and closing credits, flashbacks, news reports, and even a whole program can be repeated. Thus, a classification step is required to differentiate the P repetitions from the B ones.

3. Vocabulary Definition

Before presenting our method, in this section we define some vocabulary and some basic concepts that will help to better understand the proposed approach. First, a clear distinction should be made between the content and its representation. The "content" is somehow an abstract concept, since this content cannot exist without any representation. Each item of content as a movie or a commercial has a duration, but no start or end time, and can appear several times in the stream. Whereas the representation of this item has a start and end time, and thus it is unique in the stream. Let us define the following.

Content: what the stream represents.

Content element: a piece of contiguous content.

Content item: a content element, which corresponds to a basic broadcasting and semantic unit (e.g., a movie, a commercial, a news report, a weather forecast...). When a movie is split in two by a break, both parts are considered as content items.

Video stream: succession of frames presented at a fixed rate.

Shot: a contiguous series of frames taken by single source (camera).

Segment: any contiguous series of shots that may be semantically unrelated. A segment represents a content element.

Sequence: a contiguous series of shots that are semantically related (e.g., all the shots of a given commercial). A sequence represents a content item.

Break (B): every sequence with a commercial aim such as commercials, interludes, trailers, jingles, bumpers, and self-promotions.

Program (P): every sequence that is not a break (e.g., movies, weather forecasts, news, etc.)

A content element can appear only once in the stream or be represented several times, because it is broadcasted several times (e.g., reruns) or because it appears several times in the same program (e.g., some scenes in a cartoon). To distinguish such situations at the stream level, we will use the words: *UniqSegment* and *RepSegment*, *UniqSequence* and *RepSequence*. For example, a content item appearing several times in the stream is represented by several RepSequences in the stream.

We will call *RepSet* a set of RepSegments or RepSequences corresponding to the same content element or content item. A RepSet is a set of stream segments, which are almost identical from a content point of view. The set of all the RepSets of a stream will be called a *RepStreamSet*.

4. A Repetition-Based TV Stream Structuring Method

As we have mentioned in Section 2, the method proposed by Naturel et al. in [1] is costly and time-consuming as it uses the manual annotation to bootstrap the structuring process. Our method outperforms this approach by using a machine learning technique that limits the manual annotation to the data necessary to train the system. Such data do not need to be contiguous to the testing data. In Figure 1, we give an overview of the proposed method.

The first step consists in detecting the repetitions (i.e., the RepStreamSet). Then, a postprocessing step is applied in order to fuse the consecutive RepSets in the RepStreamSet that belong to the same sequence (e.g., same commercial) in order to get a set of RepSequences. After that, a classification method separates the program RepSequences from the break ones. Then, the P/B segmentation can be extended to the whole stream. Finally, the segmented stream is aligned with the electronic program guide (EPG) to label the various segments. Next in this section, we present each step in more detail.

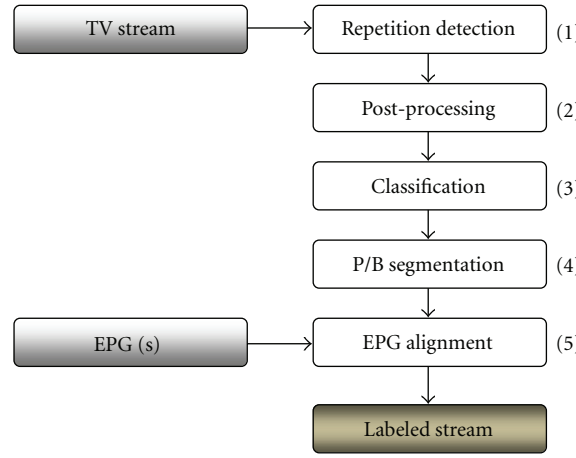

Figure 1: An overview about the proposed method (inputs: TV-stream, EPG(s). Output: labeled stream).

4.1. RepSegment Detection. The purpose of this first step is to provide a fast method for detecting the common content elements in two videos. In our context, two cases can occur (1) the element can be an item broadcasted several times or (2) it can be a part of an item that appears several times within this item. In all cases, all the representations of this element should be very similar, and the set of all possible transformations between two representations of a same content element should be rather small. These transformations come basically from noise broadcasting (additive Gaussian noise, color shift, digitization...) and from editing effects at the postproduction stage (small temporal variations, text, banners...). To consider two segments as RepSegments, these editing effects should be limited. Otherwise, the corresponding frames will present very large differences and it will not be possible to consider them as similar anymore.

The literature is very rich in methods that detect repeated sequences in the audiovisual streams. Some of these methods are limited to the repetition detection task (i.e., [17–23]) while others are designed to detect breaks and commercials (i.e., [4, 6, 13, 24, 25]). To detect repeated sequences, a simple method consists in searching in the document the frames or segments that are near-duplicate [4, 25]. Another idea is to compare the document with itself in order to provide a similarity matrix that can be processed to retrieve the repeated sequences [19]. The presented methods face problems when dealing with large audiovisual documents. For thus, Herley proposes in [20] a method that can process a continuous infinite stream. In this method, the research of repetitions is limited to a finite historic sliding window. The technique of perceptual hashing is also used to detect repetitions by considering a stream as a database containing signatures each representing a frame, a shot or, a segment [6, 13, 18, 21, 24].

As mentioned before, Naturel et al. have proposed the first complete TV stream structuring solution [1]. To our

knowledge, the results obtained by his method are the most interesting in the domain. The only claim is the use of the video reference database that needs to be annotated manually.

In our work, we do not aim to propose a method to detect repeated sequences since the literature is very rich in such methods. In addition, we try to propose a TV stream structuring solution that overcomes the limitations of the one proposed by Naturel et al. For thus, we have chosen to use the method proposed by Naturel et al. to detect repetitions. Adopting the same algorithm will also facilitate the comparison of our TV stream structuring method with the Naturel's one.

In this paper, we depend on the method presented in Naturel's work [1] to detect the RepSegment. This approach proposes a perceptual hashing technique using a reference video database (RVD) composed of manually labeled video segments. Each label contains the type of the segment (P or B) and its title. Each shot of the stream is then compared to this database. In order to avoid a time consuming exhaustive search, each frame is described by a visual signature; each shot is thus characterized by a set of signatures; the search is limited to the database shots, which have at least one common signature.

From a shot segmentation point of view, we depend on the method presented in [26]. This method uses an adaptive threshold of luminance histogram, with improvements to detect dissolves and fades. The visual signature is based on the 64 lower frequency coefficients (except the DC coefficient itself) extracted from the discrete cosine transform (DCT) of the image luminance channel. The median value of the coefficients is used to binarize these coefficients, and thus providing a 64-bit signature.

In this step, we contribute in the adaptation of the repetition detection method proposed in [1] without basing on the RVD. For thus, we propose an algorithm that compares the stream with itself in real time. First, the stream is segmented into shots, and a 64-bit visual signature is extracted from each frame. Then, these signatures are inserted in a hash table with a reference to the shot the corresponding frame belongs to. Using this representation, we can effectively compare the stream with itself to retrieve the RepSegments. Additionally, and like Naturel did, we assume that RepSegments have at least a common visual signature. Algorithm 1 shows the different steps of the detection process.

The Hamming distance is used to compare two shots and it is computed as follows.

Each shot contains several frames where each frame is assigned to a signature. Two shots p and q are compared, in case of having two frames one from p and the other from q with equal signatures. We first measure the Hamming distance between each two signatures u and v in a shot. This measurement is done by computing the number of different bits between u and v as in

$$\mathrm{Hamm}\left(\mathrm{Sig}_i, \mathrm{Sig}_j\right) = \Sigma_k \mathrm{Sig}_i[k] \oplus \mathrm{Sig}_j[k], \quad k = 1, \ldots 64. \tag{1}$$

Then, in order to measure the distance D between two shots Sh_i and Sh_j with the same duration, we take the average of the Hamming distances between the signatures of Sh_i and Sh_j as it is defined by

$$D\left(\mathrm{Sh}_i, \mathrm{Sh}_j\right) = \frac{1}{N}\left(\Sigma_k \mathrm{Hamm}\left(\mathrm{Sig}_{ik}, \mathrm{Sig}_{jk}\right)\right), \quad \text{for } k = 1 \ldots N, \tag{2}$$

where Sig_{ik} (Sig_{jk}, resp.) is the signature of the frame number k of the shot Sh_i (Sh_j, resp.) and N is the number of frames in the two shots.

In the case where the two shots have different durations, the middle frame of the first shot is aligned with the middle frame of the second one. The frames at the boundaries that do not have associated frames in the second shot are discarded when computing the distance.

Two shots are considered as having the same content if their distance D is less than a fixed threshold.

At the end, the algorithm provides sets of similar shots (or part of shots) called RepSets. Each shot is represented by its start and end times in the stream and is measured in image numbers. Most content items are composed of several shots. The detection process described above should thus be completed by a postprocessing step that combines RepSegments in order to get RepSequences. These RepSequences should correspond, if the postprocessing step was not error prone, to content items, for example, to programs (or program parts) or commercials.

4.2. Repsegments Postprocessing. The detection process explained in the previous section supplies sets of analogous stream segments, that is, RepSets of RepSegments. The goal of the next step is to glue these RepSegments together in order to get less RepSegments of longer duration. Ideally, we should get RepSequences corresponding to semantically coherent content items (programs, trailers, commercials...). From a theoretical point of view, we should fuse consecutive RepSets that have the same number. Thus, the first proposed rule deals with the simple case where two RepSets have the same number of elements, which may belong to the same repeated content (content item: same commercial, e.g.,). From a practical point of view, this rule is not sufficient. Other rules should be proposed to take into account scenarios implied by the use of some production rules by TV channels and some problems propagated by the previous steps of the TV stream structuring process (shot segmentation and repetition detection). For these reasons, we have also proposed two additional rules dealing with the fact that some segments could have been missed (not broadcasted by TV channels or missed by the shot segmentation or repetition detection steps), leading to RepSets of different cardinalities.

Let us denote R_i the various RepSets and S_i^j the segments of R_i. We assume that the S_i^j are sorted by increasing starting time, and the R_i are sorted by increasing starting time of their first element S_i^1.

The first postprocessing rule deals with the simple following case: two consecutive RepSets R_i and R_{i+1} have

(1) Compute the signatures of all frames and insert them in a hash table HT.
(2) For each signature S_i of HT,
  (a) Find in HT the set Set(S_i) of all shots where this signature appears
  (b) For all pairs $(p,\ q)$ of shots in Set(S_i)
    (i) Check if the pair is already in the list of similar shots.
    (ii) If the Hamming distance between p and q is smaller than a threshold α, insert $(p,\ q)$ in the list of similar shots.
(3) Compute and return the equivalence classes from the list of similar shots.

Algorithm 1: RepSegments detection process.

the same number of elements, and these elements are contiguous, that is, for all j, S_i^j and S_{i+1}^j are contiguous. In order to deal with some possible noise, we introduce thresholds and distances, but the idea remains the same. The fusion is straightforward in this case.

First let us define a distance between two segments. This distance is equal to the duration between the end of the earlier segment and the start of the later one if both segments were broadcasted the same day, and to $+\infty$ otherwise. Second, we extend this distance to get a comparison function ∂ between two RepSets R_i and $R_{i'}$. This function is simply the average μ and the standard deviation σ of the distances between the corresponding segments S_i^j and $S_{i'}^j$ of the two RepSets. Two RepSets are fused if μ is smaller than a first threshold and σ is smaller than a second one. In this case, a new RepSet is created whose segments are the concatenation of the S_i^j and $S_{i'}^j$ and R_i and $R_{i'}$ are removed.

We should mention here that we may fuse two consecutive RepSets even if they are of different content. This scenario appears when all the broadcasts of a commercials C_i is followed or preceded by a same commercial C_j and vice versa. In this case, the two commercials are fused in the same RepSet.

The first postprocessing rule is a simple rule that is based on the fact that two consecutive RepSets are of the same content if they have the same number of elements. This rule performs well when all the shots of a commercial are rebroadcasted and are detected as repeated. Unfortunately, this is not always the case. For thus, this rule may be sufficient from a theoretical point of view but not in practice. Some problems have appeared due to several reasons. A first reason is that a commercial may be shortened after one or several diffusions by TV channels. Another reason is due to errors in the shot segmentation and the repetition detection steps. In this case, we may have two consecutive RepSets that belong to the same content but have different number of elements. In another word, some shots in the broadcast of a commercial (e.g.) are not rebroadcasted or have not been detected as repeated. Thus we have proposed two other rules that deal with these scenarios.

The second proposed postprocessing rule corresponds to the case where two consecutive RepSets have different numbers of segment and where $|R_i| > |R_{i+1}|$. In this rule, we search if there exists a RepSet R_k with $k > i+1$ and $\partial(R_i, R_k) < (\alpha', \beta')(\partial(R_i, R_k)$ returns the mean and the standard deviation of dist. See Postprocessing-Type 1). And in this case, a new $R_{i'}$ is created as the fusion of R_i and R_k and the R_j $(j = i \ldots k)$ are removed from the RepStreamSet.

The third postprocessing rule is applied to take into account the cases when $|R_i| < |R_{i+1}|$. It consists in applying the second processing rule if the R_j $(j = i \ldots k)$ contain segments smaller than one second. In this case, some small segments are dropped, but longer segments can be obtained. At the end of this postprocessing step, we get a set of RepSets (RepStreamSet). Each segment of these sets is composed of one or several contiguous shots. In the next section, we present a procedure to separate the segments corresponding to programs from those corresponding to breaks.

4.3. P/B Classification of the Segments. The previous stages, repetition detection and postprocessing, provide sets of similar segments that we call RepSets. Each segment is represented by its start and end times in the stream. We assume that the similar segments of a given RepSet represent the same content.

The main idea of our TV stream structuring solution is to detect the boundaries of B (Breaks) segments. The latter are detected basing on their repetitive behaviors in order to obtain the RepSets. These B segments are used later to segment the stream in P/B sequences. The segmentation is achieved by extracting, in a first step, all B segments from the stream, and then considering the remaining segments longer than one minute as P ones. Unfortunately, the B segments are not the only segments that appear more than once in the stream. The content present in the RepSets can be of very diverse natures: breaks, program segments like opening and closing credits, programs broadcasted twice, small programs like weather forecast, so forth. As a consequence, the RepSets do not provide enough information to get a clear structure of the stream. Therefore, the B RepSets should be separated from P ones before segmenting the stream. To do so, a classification step should be applied to differentiate P segments from B ones.

Theoretically, we may think that the duration of the B segments is less than the P ones in the RepSets. In another word, a P RepSet may contain segments that are longer than segments in a B RepSet. This assumption is not always true in reality and duration could be insufficient for the classification. For example, the News generic in France are shorter than a lot of commercial segments. Furthermore, a

(1) For all pairs (R_i, R_{i+1}) of successive RepSets
  (a) If R_i and R_{i+1} have the same number of elements n
    (i) For j ranging from 1 to n
      $\text{dist}[j] = S_{i+1}^{j} \cdot \text{start\_time} - S_i^j \cdot \text{end\_time}$
    (ii) $\partial(R_i, R_j) = (\text{mean(dist)}, \text{standard\_deviation(dist)})$
      If $\text{mean(dist)} < \alpha$ and $\text{standard\_deviation(dist)} < \beta$
        Insert $(i, i+1)$ in the result
(2) Return the result

Algorithm 2: Post-processing—Type 1.

repeated P segment may not be entirely detected or may be oversegmented. This can be due to a shot segmentation problem or a repetition detection problem which lead to obtain several shorter P segments instead of the entire P one. Moreover, the detection of false-positive repeated segments in TV programs such as talk-shows may oversegment the TV stream when considering them as B segments. In such programs, several static long-views segments with no-motion could have approximately the same duration, the same content, and could be filmed by the same camera which may mislead the repetition detection process.

To separate the P RepSets from B ones, two different methods are proposed according to the data to be classified.

(i) In the first method, each segment of each RepSet in the RepStreamSet is described by a set of local and global features and then classified. In this case, the data to be classified are the segments. This approach is noted as segment-based. The segments may be of P or B type. This method is important to classify the RepSets that contain segments of different types (P and B segments in the same RepSet) because each segment is classified independently.

(ii) The second method classifies directly each RepSet in the RepStreamSet rather than the individual segments. Each RepSet is described by a list of global features. This approach is noted as RepSet-based. Most RepSets contains only P segments or only B segments. Trailers present a third quite annoying case. A RepSet corresponding to a trailer contains several B segments representing the diffusion of trailer itself and one or several P segments corresponding to the announced program. Labeling the RepSet as B will over-segment the corresponding program. Consequently, the classification step should separate the data in three rather than two classes: (1) P RepSets, (2) B RepSets, (3) Trailer (T) RepSets.

4.3.1. Segment and Repset Features. To classify the segments and the RepSets, a set of features are proposed to describe them. Some features are chosen based on observations that we have acquired in our real life when watching television (i.e., most of breaks repeats more than once). Others are derived from rules used by TV channels (i.e., presence of silence segments before commercials). Other features may be useful in this classification (features derived from audio repetitions). In this section, we present the features used to represent the segments and the RepSets.

Segment Features. To describe a segment S_i^j, a set of global and local features is used. The local features are issued from the neighboring segments. The global ones are issued from the RepSet R_i, which contains the segment. The global features used to represent a segment are the following.

(i) $(|R_i|)$: Number of occurrences of S_i^j. It represents how many times the corresponding content item was broadcasted.

(ii) Number of days: this feature counts the number of different calendar days where the corresponding content item appears.

(iii) Number of days of the week: it measures the number of different days of the week (Monday to Sunday) where the content appears. For example, for a content item broadcasted 10 consecutive Tuesdays, the number of days will be 10 while the number of days of the week will be 1.

(iv) Duration of the segment S_i^j.

The local features used to describe a segment are issued from two sources. The first source is the presence of a separation before and/or after a segment. We call separation the simultaneous occurrence of monochrome frames and silence that happens between commercials in France due to legal regulations. To detect monochrome frames, we use the method proposed by [1]. In this method, a 48-bin histogram on the luminance channel is computed and its entropy is thresholded. The entropy of a histogram h quantized into n bins is given by

$$H = -\sum p_i \log p_i\,, \qquad p_i = \frac{h(i)}{\Sigma_k h(k)}\,. \tag{3}$$

On the other hand, the silence segments are detected using a simple method where the log-energy of overlapping audio frames of 10 ms is computed using the standard formula

$$E_{db}(i) = 10\log_{10}\Sigma_n {x_n}^2(i). \tag{4}$$

A successive analysis is used to merge silent segments and monochrome frames. The audio feature being more discriminative than the visual one, silent audio segments

Table 1: Separation detection results.

Modality	Precision	Recall
Audio only	0.82	0.9
Image only	0.41	0.89
Successive analysis	1	0.9

are detected first, and their correspondence to monochrome frames is then checked. Table 1 presents the results obtained by Naturel et al. in [1].

The separations are used as a binary feature. For each segment, it tells whether a separation was observed in a temporal window before and/or after it.

This type of features is helpful to differentiate between P and B segments. A separation may appear before and after Breaks. Such separations do not appear within programs, but can appear at their borders, before the opening credits or after the closing credits, and thus separate them with breaks. These separations do not follow a strict and systematic rule, and systems relying only on such information can produce poor results according to the production rules of the considered channel.

The second source of information to describe locally a segment is its neighboring segments. Let S_i^j be a RepSegment. First, we consider a temporal window W^b before S_i^j, and the RepSegments $S_k^b(S_i^j)$ contained in W^b. We count their number $N^b(S_i^j)$. Each of these RepSegment $s_k^b(S_i^j)$ belongs to a RepSet whose cardinality is denoted $C_k^b(S_i^j)$. This is the number of times the content of $s_k^b(S_i^j)$ was broadcasted. Second, we consider the same quantities relative to a temporal window W^a after S_i^j: $N^a(S_i^j)$ and $C_k^a(S_i^j)$. From the above information, we derive the following local features:

(i) three binary features are issued from the separations: (1) presence of a separation before S_i^j; (2) presence of a separation after S_i^j; and (3) presence of separations before and after S_i^j;

(ii) two other features are $N^b(S_i^j)$ and $N^a(S_i^j)$;

(iii) five features are issued from the $C_k^b(S_i^j)$ and $C_k^a(S_i^j)$:

(1) $\mathrm{Sum}^b(S_i^j) = \Sigma_k C_k^b(S_i^j)$,

(2) $\mathrm{Sum}^a(S_j^i) = \Sigma_k C_k^a(S_i^j)$,

(3) $\mathrm{Min}(S_i^j) = \min(\mathrm{Sum}^b(S_i^j), \mathrm{Sum}^a(S_i^j))$,

(4) $\mathrm{Max}(S_i^j) = \max(\mathrm{Sum}^b(S_i^j), \mathrm{Sum}^a(S_i^j))$,

(5) $\mathrm{Avg}(S_i^j) = \mathrm{average}(\mathrm{Sum}^b(S_i^j), \mathrm{Sum}^a(S_i^j))$.

Repset Features. Two kinds of global features are used to describe a set of repeated segments corresponding to a same content, that is, a RepSet R_i. The first ones are analogous to those used to describe segments:

(i) $|R_i|$,

(ii) number of days,

(iii) number of days of the week,

(iv) mean duration of the segments in R_i.

The second ones come from the local features defined for segments.

(i) Percentage of segments of R_i that have (1) a separation before, (2) a separation after, (3) a separation before or after, and (4) a separation before and after them.

(ii) $\sum_j N^b(S_i^j)$ et $\sum_j N^a(S_i^j)$.

(iii) For all the segments S_i^j of R_i, we compute $\mathrm{Sum}^b(S_i^j)$, $\mathrm{Sum}^a(S_i^j)$, $\mathrm{Min}(S_i^j)$, $\mathrm{Max}(S_i^j)$, and $\mathrm{Avg}(S_i^j)$. We then associate 13 features to R_i:

(a) $\min_j(\mathrm{Sum}^b(S_i^j))$, $\max_j(\mathrm{Sum}^b(S_i^j))$, $\mathrm{average}_j(\mathrm{Sum}^b(S_i^j))$,

(b) $\min_j(\mathrm{Sum}^a(S_i^j))$, $\max_j(\mathrm{Sum}^a(S_i^j))$, $\mathrm{average}_j(\mathrm{Sum}^a(S_i^j))$,

(c) $\min_j(\mathrm{Min}(S_i^j))$, $\max_j(\mathrm{Min}(S_i^j))$,

(d) $\min_j(\mathrm{Max}(S_i^j))$, $\max_j(\mathrm{Max}(S_i^j))$,

(e) $\mathrm{average}(\mathrm{Avg}(S_i^j))$.

As a result, we can say that fourteen features, four global and ten local, describe a segment, while twenty three features describe a RepSet.

4.4. Stream Segmentation. The previous step was devoted to the P/B classification of the RepSets. The goal of the present step is to segment the stream in P/B sequence. It is composed of several substeps. At the first substep, all the segments that are classified as breaks are retrieved from the stream. At this moment, the stream is segmented in presegments where each has a start time and an end time.

Let $\mathrm{Stm} = \{\mathrm{Seg}_i/\mathrm{Seg}_i = (\mathrm{Seg}_i \cdot \mathrm{start\_time}, \mathrm{Seg}_i \cdot \mathrm{end\_time})\}$ represents the segmented stream.

The aim of the second substep is to classify the remaining segments of the stream (each segment of Stm) as being a P or a B segment. The classification is based on the length of segments. A fixed threshold $d_{\min}$ is used. Every segment longer than $d_{\min}$ is labeled as P and all the other ones as B. Experiments were done by Naturel et al. [1]] and led them to fix the threshold to one minute.

In the third substep, we fuse the consecutive B segments into one segment. At the end of this step, we obtain a new segmentation of the stream as P/B sequence.

4.5. Segment Labeling. Once the stream is segmented, the next step is to add a label to each program segment. Two types of analysis methods may be used here to label the segmented stream: content-based analysis or metadata analysis. In our work, we have used the metadata broadcasted with the stream, namely, the EPG (Electronic Program Guide). The EPG contains useful information about the programs

Table 2: Segment classification using cross-validation sampling method.

CV 10-folds	Programs		Breaks	
	Precision	Recall	Precision	Recall
Random forest	96.38%	98.01%	97.66%	95.76%
Classification Tree	97.29%	97.48%	97.09%	96.87%
C4.5	97.12%	97.29%	96.88%	96.68%
KNN (10NN)	95.98%	96.66%	96.12%	95.33%
Naïve Bayes	92.36%	94.93%	93.97%	90.95%
SVM (RBF)	92.31%	95.54%	94.65%	90.83%
CN2	95.77%	98.34%	98.03%	95.00%

Table 3: Segment classification using random sampling method (30% to train and 70% to test).

RS (30 : 70) iterated 5 times	Programs		Breaks	
	Precision	Recall	Precision	Recall
Random forest	96.17%	97.08%	96.59%	95.55%
Classification Tree	96.29%	96.76%	96.25%	95.71%
C4.5	96.46%	97.03%	96.56%	95.89%
KNN (10NN)	95.16%	96.19%	95.56%	94.36%
Naïve Bayes	92.41%	94.88%	93.92%	91.02%
SVM (RBF)	94.37%	96.44%	95.79%	93.38%
CN2	95.31%	97.71%	97.28%	94.46%

broadcasted, for example, titles, genres, and sometimes other information such as a short description and the list of actors.

To label the segmented stream, we propose to align it with the EPG using the Dynamic Time Warping (DTW) algorithm. DTW is a well-known method that computes a path and a distance between two sequences X and Y. The distance may be interpreted as the cost to be paid to transform X into Y by a set of weighted edition operations. These operations are the insertion, deletion, and substitution. The path with minimal cost provides the best alignment. In our system, a distance is computed between segments. It measures the similarity of the segments in terms of durations, start, and end times.

$$\begin{aligned}\mathrm{dist}\left(\mathrm{seg}_i, \mathrm{seg}_j\right) = {} & \left|\mathrm{duration}_i - \mathrm{duration}_j\right| \\ & + \left|s_i - s_j\right| + \left|e_i - e_j\right|,\end{aligned} \tag{5}$$

where $\mathrm{duration}_i$ ($\mathrm{duration}_j$, resp.) is the duration of seg_i (seg_j, resp.), s_i (s_j, resp.) is the start time of seg_i (seg_j, resp.) and e_i (e_j, resp.) is the end time of seg_i (seg_j, resp.).

The cost of the insertion deletion and substitution operations are defined as

$$\begin{aligned} C_{\mathrm{del}} &= \mathrm{dist}\left(\mathrm{seg}_i, \mathrm{seg}_j\right), \\ C_{\mathrm{ins}} &= \mathrm{dist}\left(\mathrm{seg}_i, \mathrm{seg}_j\right), \\ C_{\mathrm{sub}} &= \alpha * \mathrm{dist}\left(\mathrm{seg}_i, \mathrm{seg}_j\right), \quad \text{where } 1 < \alpha < 2. \end{aligned} \tag{6}$$

The α parameter is used to favor a substitution operation over a deletion followed by an insertion one. Reader can refer to Naturel's Ph.D. [15] for more information.

5. Experiments

This section presents a series of experiments to illustrate the results of applying our method. In these experiments, we used a corpus of 22 consecutive days of television recorded from a French channel (France2), and this is for the duration from 9/5/2005 to 30/5/2005. The evaluation of the proposed method concerns the classification, the segmentation, and the alignment steps. The repetition detection and postprocessing steps are not considered for two reasons. The first is the practical impossibility to manually annotate a database in terms of repeated content. Some methods in the literature provide an estimation of the precision of the repetition detection process. They choose randomly a sample set of repetitions and verify them manually. This type of evaluation remains imprecise. The second reason is that we would like to compare our results to the one proposed by Naturel et al. and, as a consequence, we adopt approximately the same evaluation process.

5.1. Classification Evaluation. The first experiments evaluate the RepStreamSet classification since it is the core of our contribution. The first set of these experiments are at segment level (segment-based) where each segment is described by its 14 features and classified as a P or B segment. On the other hand, the second part of experiments is at the RepSet level (RepSet-based), where each RepSet is described by its 23 features and classified as P RepSet, B RepSet, or T(= Trailer) RepSet. In this section, we present the experiments at the two levels, and then we compare the results issued from both levels.

Several classification methods were compared (Random Forest, Classification Tree, C4.5, KNN, Naïve Bayes, SVM,

Table 4: RepSet classification using cross-validation sampling method.

CV 10-folds	Trailers	Programs		Breaks	
	Precision	Precision	Recall	Precision	Recall
Random forest	62.16%	31.08%	97.75%	98.19%	93.24%	93.49%
Classification Tree	43.14%	29.73%	98.13%	97.45%	91.31%	94.20%
C4.5	37.20%	23.46%	97.65%	97.52%	91.35%	92.83%
KNN (10NN)	42.11%	21.62%	97.621%	96.39%	88.24%	93.04%
Naïve Bayes	0.89%	78.38%	99.89%	13.08%	95.29%	73.10%
SVM (RBF)	72.73%	10.81%	97.88%	97.12%	90.00%	94.56%
CN2	64.86%	32.43%	97.02%	97.94%	92.82%	91.66%

Table 5: RepSet classification using random sampling method.

RS (30 : 70) iterated 5 times	Trailers		Programs		Breaks	
	Precision	Recall	Precision	Recall	Precision	Recall
Random forest	62.50%	13.46%	97.85%	97.82%	91.87%	94.34%
Classification Tree	29.91%	13.46%	98.00%	97.23%	90.49%	94.28%
C4.5	36.54%	25.68%	97.72%	97.77%	92.17%	92.95%
KNN (10NN)	28.57%	9.23%	97.41%	96.05%	87.15%	92.83%
Naïve Bayes	0.92%	76.54%	99.94%	35.23%	93.96%	77.69%
SVM (RBF)	72.73%	10.81%	97.88%	97.12%	90.00%	94.56%
CN2	27.16%	8.46%	97.89%	97.07%	90.22%	94.59%

CN2), and several sampling methods were used to validate our results using the orange data mining software [27] (cross validation noted CV, random sampling noted RS). The corpus was manually annotated and then used to label the RepSegments and RepSets in the RepStreamSet. The labeled RepStreamSet is used to train all the tested classification methods on part of the data from the one hand, and to check the results on the remaining part from the other hand.

5.1.1. Segment Classification. The set of labeled RepSegments in the RepStreamSet is used to evaluate the segment-based classification approach. As we have already seen, each segment can take one of two labels: P segment or B segment. It should be noticed that, due to postprocessing errors, a segment could aggregate both P and B shots. In such a case, the segment is considered P or B in the ground truth according to the most overlapping class. This case rarely occurred in our results. Tables 2 and 3 show the precision and recall using several classification and sampling methods.

5.1.2. RepSet Classification. The set of labeled RepSets in the RepStreamSet is used to evaluate the RepSet-based classification approach. Contrarily to segment classification, RepSets can take one of the three labels: purely P RepSet, purely B RepSet, or T(= trailer) RepSet. Tables 4 and 5 show the precision and recall obtained with the same classification and sampling methods as above.

We can observe the poor detection results of T RepSets. This may be due to several problems. First, the small number of trailers in the corpus is not sufficient to train correctly the corresponding model (178 trailers in a set of 14280 RepSets). Second, the pattern of these RepSets is much more complex mixing P and B segments. Third, we faced some incoherencies in the shot detection process. Some content items that are broadcasted several times were not segmented always in the same way. This may be due to capturing problem, which has affected the quality of the analogical streams that have been captured.

In order to correct the missclassification of the T RepSets, a priori filtering rule is applied in order to separate most of the T RepSets before the classification step. This rule is based on the local features of the segments contained in the RepSet. For example, let R_i be a T RepSet. The P segments of R_i will not have neighboring RepSegments (before or after). In other words, a program may have isolated RepSegments when this does not happen with B sequences.

We have also used two other rules to filter some P RepSets. The first rule detects the RepSets with which the mean duration is greater than three minutes. These RepSets take a P label since B sequences are overwhelmingly shorter. We have also observed that in programs such as political debates, games, and talk shows, some shots are detected as repeated within a given program. Such detections are due to the fact that in these programs some shots are taken with a fixed camera, have the same duration, and the scene is very static with no motion. As a consequence, these shots are almost identical from a visual point of view and give raise to RepSets. Other shots may also be detected like opening and closing credits when the program is split by a break, or shots to show the presents to win in game programs. The second rule uses the proximity of such segments in the same RepSet and the ratio of separations before and after these segments.

Applying the three previous rules on the RepSets provides two RepStreamSets as result. The first called, for simplicity, RepStreamSet_filtered which represents the

Table 6: Classification of the RepSets after filtering using cross-validation sampling method.

CV 10-folds	Trailers		Programs		Breaks	
	Precision	Recall	Precision	Recall	Precision	Recall
Random forest	31.51%	32.86%	95.85%	95.08%	91.32%	92.45%
Classification Tree	34 %	24.29%	94.82%	94.88%	90.49%	91.06%
C4.5	36.51%	32.86%	95.55%	95.68%	91.96%	92%
KNN (10NN)	58.54%	34.29%	96.38%	94.35%	90.14%	94.32%
Naïve bayes	2.32%	70%	99.60%	66.03%	94.11%	75.92%
SVM (RBF)	90%	12.86%	96.07%	94.77%	90.05%	94.13%
CN2	20.78%	22.85%	94.82%	94.5%	90.42%	90.69%

Table 7: Classification of the RepSets after filtering using random sampling method.

RS (30 : 70) iterated 5 times	Trailers		Programs		Breaks	
	Precision	Recall	Precision	Recall	Precision	Recall
Random forest	20.49%	20.58%	95.31%	94.80%	91 %	91.84%
Classification Tree	19.15 %	22.04%	94.57%	94.70%	90.57%	90 %
C4.5	26 %	24.08%	94.92%	94.87%	90.51%	90.77%
KNN (10NN)	29.33%	26.94%	94.35%	93.50%	88.55%	90.09%
Naïve bayes	2.37%	68.98%	99.55%	67.88%	93.91%	75.29%
SVM (RBF)	20.49%	20.58%	95.31%	94.80%	91%	91.84%
CN2	0%	0%	97.20%	94.24%	89.21%	96.11%

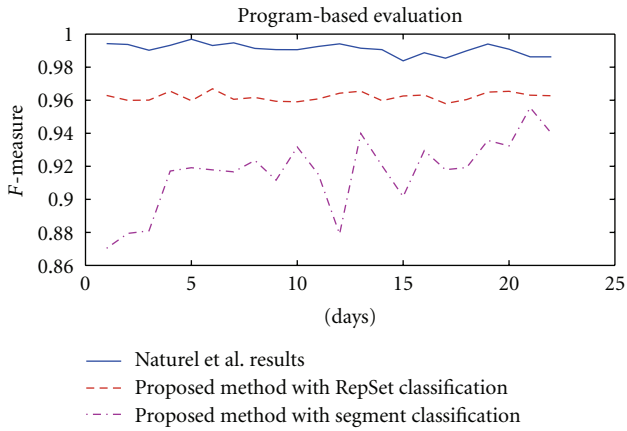

Figure 2: Results in terms of P F-measure.

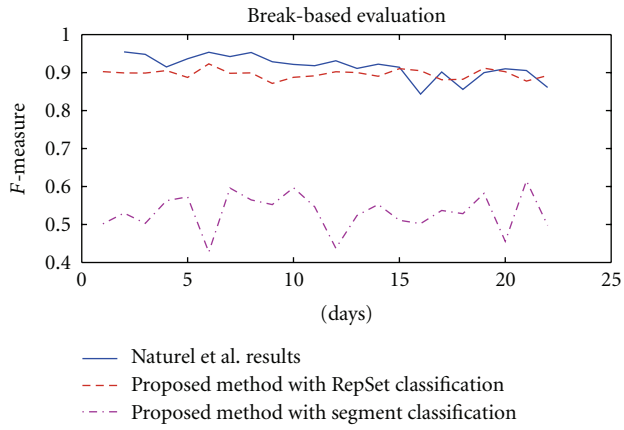

Figure 3: Results in terms of B F-measure.

RepSets that obey the filters. The second called RepStreamSet_remained that represents the remaining RepSets after the filtering step. Returning to the experiments, 5114 RepSets are filtered (i.e., the size of RepStreamSet_filtered), while 9166 are the remaining ones (i.e., size of the RepStreamSet_remained). In RepStreamSet_filtered, 5025 P RepSets and 88 Trailer RepSets are correctly filtered.

After the application of the three filters mentioned above, a new set of repeated content is used (i.e., RepStreamSet_remained). Tables 6 and 7 show the precision and the recall results using the new set of repeated content (RepStreamSet_remained).

Most of the RepSets in the RepStreamSet_filtered are filtered correctly. They were not considered when computing the precision and the recall of the classification. By adding the correctly filtered RepSets to the previous tables, we obtain new results (Tables 8 and 9). These results give better performances, especially for the T RepSets.

As complementary results, we make an additional experiment to check whether the performance of the classification at the segment level remains the same when using the two filters or not. Tables 10 and 11 show the precision and the recall of the previously used methods using the cross-validation sampling method, but at the segment level.

5.1.3. Comparing Segment-Based and Repset-Based Classification. In this section, we have chosen randomly 30% of the RepSets to train the models and then all the RepSets are classified in order to associate all the data to a class even if it belongs to the training set. Tables (12(a), 12(b), 12(c),

Table 8: Table 6 + counting the filtered RepSets.

CV 10-folds + filtering	Trailers		Programs		Breaks	
	Precision	Recall	Precision	Recall	Precision	Recall
Random forest	71.69%	77%	98%	98.24%	91.32%	92.14%
Classification Tree	77%	74%	97.48%	98.14%	90.49%	90.76%
C4.5	75.12%	77%	97.83%	98.51%	91.96%	91.69%
KNN (10NN)	84.5%	77.45%	98.23%	97.89%	90.14%	94%
Naïve bayes	8,12%	89.7%	99.75%	84.53%	94.11%	75.66%
SVM (RBF)	91.66%	70.1%	98%	97.43%	90.05%	93.81%
CN2	67,26%	73,5%	97,5%	97,4%	90,42%	90,38%

Table 9: Table 7 + counting the filtered RepSets.

RS (30 : 70) iterated 5 times + filtering	Trailers		Programs		Inter-program	
	Precision	Recall	Precision	Recall	Precision	Recall
Random forest	78.62%	78.46%	98.12%	97.97%	91 %	91.39%
Classification Tree	76 %	78.87%	97.83%	97.93%	90.57%	89.57 %
C4.5	80.88 %	79.38%	97.96%	98%	90.50%	90.33%
KNN (10NN)	82.55%	80.10%	97.75%	97.48%	88.55%	89.65%
Naïve bayes	11.25%	90.67%	99.75%	87.69%	93.9%	74.93%
SVM (RBF)	96.61%	76.20%	98.17%	98%	90.22%	92.5%
CN2	97.41%	73.33%	98.85%	97.76%	89.22%	95.65%

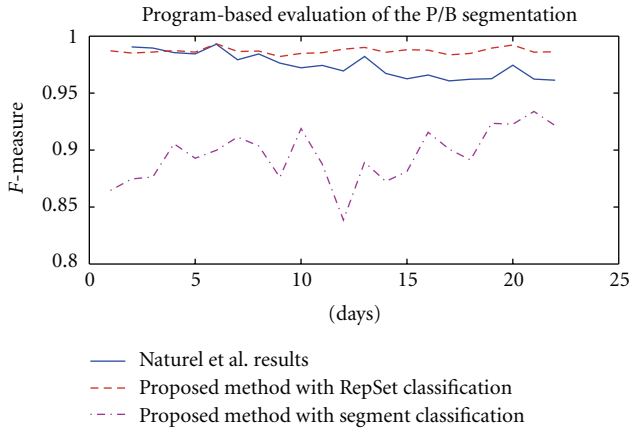

Figure 4: Results in terms of P F-measure, without separations.

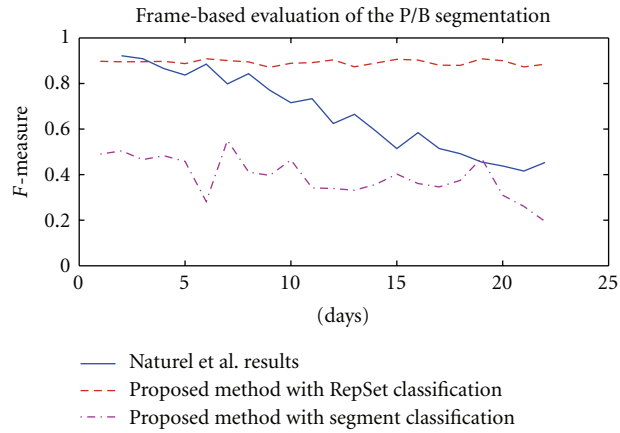

Figure 5: Results in terms of B F-measure, without separations.

and 12(d)) show the distribution of RepSets between classes using several classification methods. The lines correspond to the true class of each RepSet, and the columns to the class in which these RepSets were classified.

In an analogous way, we have chosen randomly 30% of the RepSegments and classified the whole set. Tables (13(a), 13(b), 13(c), and 13(d)) show the confusion matrices.

In order to compare the results obtained with RepSets to those obtained with the segments, the former should be translated in terms of segments. To this aim, we compute the percentage of well-classified segments that are produced by the RepSet classification. Table 14 gathers the results in terms of number of RepSets (column 1) or segments (columns 2 and 3) correctly classified. Columns 1 and 3 correspond to the classification of the RepSets and the segments, respectively. Column 2 is the translation of column 1 in terms of correctly classified segments (only columns 2 and 3 can be directly compared).

It should be noticed that the random forest algorithm provides the best performances and appears as the most adapted algorithm to our problem. Furthermore, the comparison of the two numbers in boldface shows that RepSet-based classification provides a better performance than segment-based classification. On the other hand, the rate of misclassification falls from 5.27% to 3.28%. These results mean that using information about all the repetitions of a content item rather than just considering the RepSegments in their local context only is useful and increases the global performance of the system.

Table 10: Classification of the segments after filtering using cross-validation sampling method.

Cross-validation10-folds	Programs		Breaks	
	Precision	Recall	Precision	Recall
Random forest	96.56%	97.14%	93.91%	92.72%
Classification Tree	97.44%	97.23%	94.20%	94.63%
C4.5	97.26%	97.45%	94.63%	94.23%
KNN (10NN)	96.05%	96.81%	93.18%	91.63%
Naïve bayes	92.93%	92.48%	84.35%	85.23%
SVM (RBF)	94.17%	96.94%	93.15%	87.38%
CN2	96.59%	98.03%	95.72%	92.73%

Table 11: Table 10 + counting the filtered segments.

Cross-validation10-folds + filtering	Programs		Breaks	
	Precision	Recall	Precision	Recall
Random forest	97.79%	98.18%	94.58%	93.43%
Classification Tree	98.35%	98.23%	94.82%	95.15%
C4.5	98.17%	98.25%	94.84%	94.62%
KNN (10NN)	97.47%	97.97%	93.40%	92.52%
Naïve bayes	95.53%	95.26%	86.17%	86.90%
SVM (RBF)	96.26%	98.05%	94%	88.79%
CN2	97.8%	98.73%	96.17%	93.48%

Table 12

(a) Classification Tree.

B	19	122	2101
P	5	6733	113
Trailers	28	20	26
	Trailers	P	B

(b) Random Forest.

B	10	98	2134
P	1	6760	90
Trailers	34	17	23
	Trailers	P	B

(c) SVM.

B	2	109	2131
P	0	6601	250
Trailers	8	25	41
	Trailers	P	B

(d) KNN (10NN).

B	8	230	2107
P	8	6613	230
Trailers	14	27	33
	Trailers	P	B

Table 13

(a) Classification Tree.

B	2228	20329
P	25048	939
	P	B

(b) Random Forest.

B	1829	20728
P	25259	728
	P	B

(c) SVM.

B	3291	19266
P	25066	921
	P	B

(d) KNN (10-NN).

B	1829	20728
P	25259	728
	P	B

5.2. Stream Segmentation and Characterization. The goal of the present section is to enlarge the classification step to the whole stream, that is, to UniqSegments also, and to label the segments whenever this is possible. As it is mentioned before, this labeling stage is done by aligning the stream with an electronic program guide (EPG) provided with the stream or by an external source. In order to compare our solution to the one proposed in [1], we have adopted the same algorithm that segments the stream in P/B segments (as described above) and aligns this segmentation with the EPG, with the same parameters. As it was done in [1], we tried to use or not the detection of separations (silence + monochrome frames) in order to measure the influence of this feature.

Table 14: Comparison between RepSet-based and segment-based classification.

Classification tree	96.67%	95.57%	93.5%
Random forest	97.4%	96.72%	94.73%
SVM	95.34%	94.94%	91.32%
KNN	95.28%	94.09%	92.31%
Classification	RepSet-based	RepSet-based in terms of segments	Segment-based

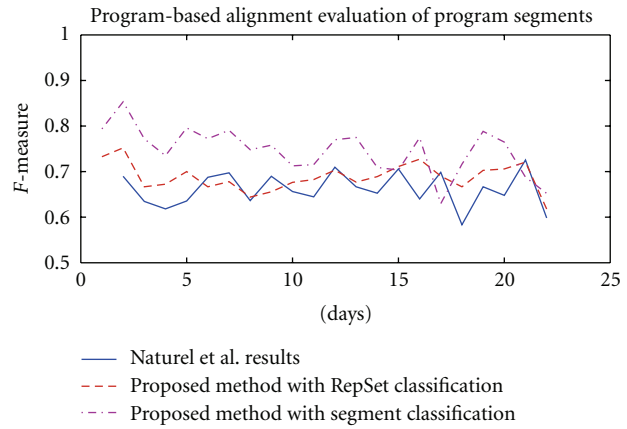

Figure 6: Alignment evaluation in terms of program segments with separation information.

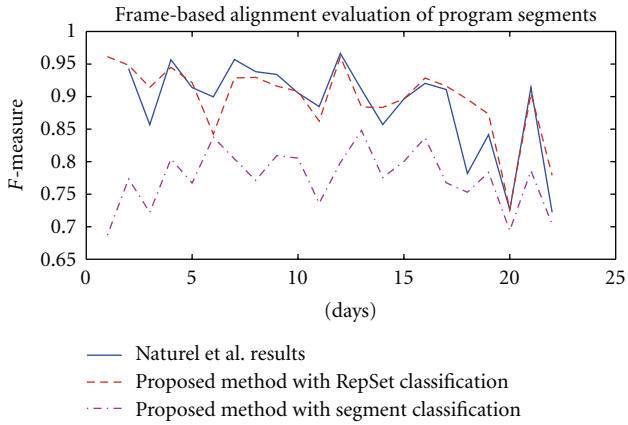

Figure 7: Alignment evaluation in terms of program frames with separation information.

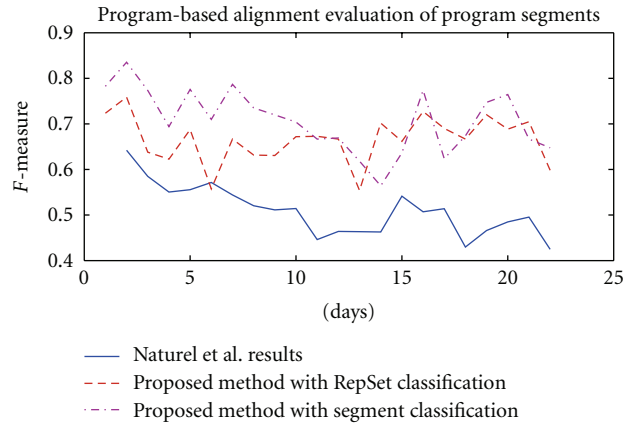

Figure 8: Alignment evaluation in terms of program segments without separation information.

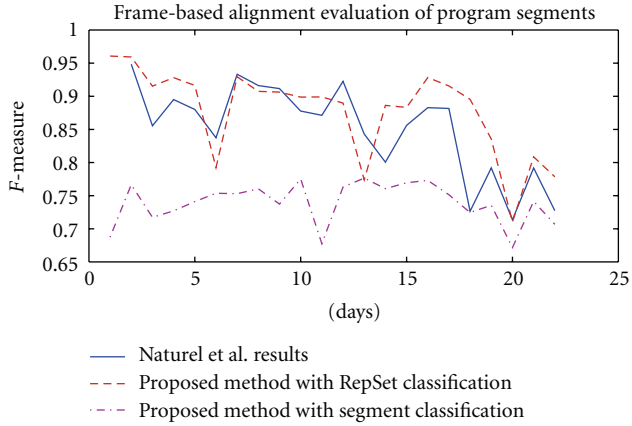

Figure 9: Alignment evaluation in terms of program frames without separation information.

Three reasons pushed us to compare our results to the ones obtained by Naturel et al. The first is that our method is based on some techniques of their work while it has also worked to overcome the drawbacks. The second reason is that to our knowledge, their results can be considered the best obtained results in the stream structuring area. The third reason is that we could use the same data and the same ground truth. For more details about the algorithm and the used parameters, reader can refer to Naturel's Ph.D. [15]. The rest of this section is devoted to present the segmentation evaluation, and then the alignment and the labeling evaluation.

5.2.1. Segmentation Evaluation. Although the problem is a segmentation one, we view it as a binary classification problem (in P/B classes) in order to facilitate the evaluation. This allows the use of the classical precision and recall measures at the frame level, but it requires choosing one of the two classes. We choose to use the program segments as they can be considered the most interesting for users. The problem is that the majority of the stream is composed of program frames, and this may bias the results since classifying the full stream as P will provide good results. That is why we also evaluate our system by computing the precision and recall relative to the B class.

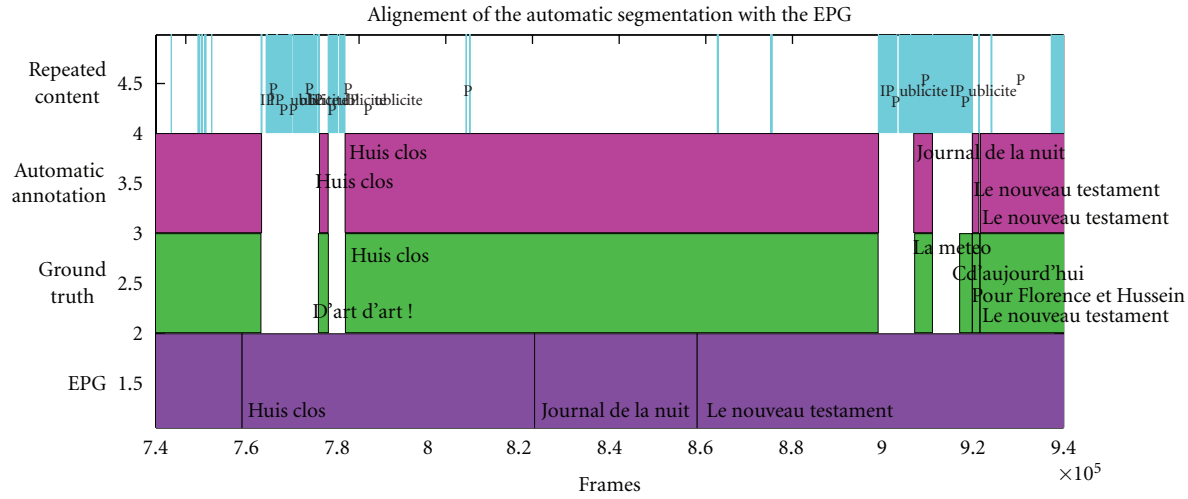

Figure 10: Portion of the alignment results of the first corpus day with its associated Electronic Program Guide.

For the P class, the precision is the ratio of the number of P-frames classified as P to the number of frames classified as P, when the recall is the ratio of P-frames classified as P to the total number of P-frames. The same quantities are defined for the B class.

$$\begin{aligned}
\text{P-Precision} &= \frac{(\text{no. of P-frames classified as P})}{(\text{no. of frames classified as P})},\\
\text{P-Recall} &= \frac{(\text{no. of P-frames classified as P})}{(\text{no. of P-frames})},\\
\text{B-Precision} &= \frac{(\text{no. of B-frames classified as B})}{(\text{no. of frames classified as B})},\\
\text{B-Recall} &= \frac{(\text{no. of B-frames classified as B})}{(\text{no. of B-frames})}.
\end{aligned} \tag{7}$$

In order to combine these measures, we use the classical F-measure:

$$F\text{-measure} = \frac{(2 * \text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}. \tag{8}$$

Figures 2 and 3 show the segmentation results obtained for each day of the stream in terms of P and B F-measures. In these experiments, the separation information is used in the segmentation process. The blue curve corresponds to Naturel's results, the brown one represents our method based on RepSet classification and the pink one represents our method based on segment classification.

The segmentation done at the segment level appears not to be stable over days and provides worse results. The global information used in the RepSet classification is the key element, which explains the difference with the segmentation based on the local information. With the P F-measure, Naturel's method has a score between 98 and 99% and outperforms ours that has a score of about 96%. On the other hand, in terms of B F-measure, Naturel's method and our method give almost the same results. It can be noticed that the results of Naturel's method tends to decrease slightly over time. This is probably because of the depreciation of the reference video database used in this method. Only few of the breaks of this database are still repeated in the stream after 20 days. The results of our method are more stable, and this proves the efficiency of our choice to rely on repetitions detection rather than on a manually annotated reference set of videos.

However, our method still show some weaknesses, partly due to the shot segmentation program. From a theoretical point of view, a RepContent should be segmented using the same way in each time it appears in the stream. In other words, a commercial that appears several times in the stream should be segmented by the same way. This was not the case in practice, and it happened that two representations of a same content were segmented differently. As we have already mentioned, the cause of this limitation can be the video capturing process, which is affected by several broadcasting and capturing effects. To avoid this problem, Naturel et al. allow the comparison of shots of different lengths in the RepSegment detection algorithm. The best alignment that provides the lowest distance is searched when comparing two shots. This feature is not implemented yet in our technique.

Figures 4 and 5 show the results obtained without using the separations in the classification process. We can notice the importance of the separation information in Naturel's method. The depreciation of the reference database over time is not corrected by the separation detection anymore, and the results decrease rapidly. On the other hand, our method is not affected by the loss of this source of information and our results stay stable over days.

5.2.2. Alignment Evaluation. Two measures are used to evaluate the performance of the alignment process. The first one uses the program segments as basic temporal units for the evaluation. The second one considers the same program segments but at the frame level. Both use the *F-measure* computed from the precision and recall measures: the precision is the ratio of the number of program segments

(resp. program frames) annotated correctly by the system over the total number of program segments (resp., program frames) that were annotated by the system. In its turn, the recall is the ratio between the number of program segments (resp., program frames) annotated correctly by the system and the total number of program segments (resp., program frames) present in the corrected EPG (ground truth EPG).

Figures 6, 7, 8, and 9 show the alignment evaluation in terms of F-measure computed, respectively, with program segments and programs frames with separation information or not.

As shown in Figures 6 and 7, the performance of our method is higher than those obtained by Naturel's.

When using the frame as basic unit for evaluation, the curves of the annotation step generated by Naturel's method and by ours are quasi-identical, especially when the classification is done on the RepSets.

One of the encountered problems during the alignment process comes from the fact that small program segments are not announced in the EPG. Our method detects the start and the end of these programs but their label cannot be predicted since it is not presented in the EPG. Most of the time, the alignment algorithm does not add labels to these segments. The use of a manually annotated database as used in Naturel's work may somehow overcome this problem since the database contains some of these programs. Figure 10 shows an example of such a problem. As we can see, the "Dart d'art" program segment is not announced in the EPG. However, its accurate start and end were correctly detected and the correct P label has been assigned to the segment. But it was finally erroneously annotated, due to the lack of the correct annotation in the EPG. The same phenomenon occurs with "La meteo" segment which is annotated as "Journal de nuit" since the first label is not present in the EPG.

6. Conclusion

In this paper, we have proposed a new TV stream structuring method that overcomes the drawbacks in the initial work of Naturel et al. The first step of our method was to detect repeated contents in the stream. This detection is done automatically with no use of any a priori annotated database. Then, a classification step is applied to separate the programs from the breaks. In its turn, the classification is achieved at the segments level or at the level of the sets of repeated segments, using a mixture between local and global features. Once classified, the segments serve to segment and classify the remaining part of the stream. Finally, the segmented stream is aligned with the electronic program guide (EPG) in order to propagate the program labels (their title most of the time.). For evaluating our method, we used the same corpus as Naturel's and our results proved the efficiency of the proposed solution.

During the experiments, we faced a major problem concerning the piece of stream that we use. The first step of the method is the detection of all repeated shots of the stream. In order to avoid the weaknesses of a given shot detector, we tested several of them. Nevertheless, none of these methods was stable enough, and all could sometime segment differently two diffusions of the same content item. As a consequence, some of the repetitions could not be detected. We think that this problem occurs because of the used database capturing method. The capture of the database was done by independent slices of 24 hours on a broadcast signal, which was not digital. That means, the capturing method may differ from a day to another due to weather circumstances or other reasons. An important future step will be to study the improvement that can be added to our experiences in case of using them with a stream of a digitally broadcasted signal.

References

[1] X. Naturel, G. Gravier, and P. Gros, "Fast structuring of large television streams using program guides," in *Proceedings of the 4th International Workshop on Adaptive Multimedia Retrieval*, pp. 223–232, Geneva, Switzerland, March 2006.

[2] G. Manson and S. A. Berrani, "Automatic TV broadcast structuring," *International Journal of Digital Multimedia Broadcasting*, vol. 2010, 2010.

[3] J. P. Poli, "An automatic television stream structuring system for television archives holders," *Multimedia Systems*, vol. 14, no. 5, pp. 255–275, 2008.

[4] P. Duygulu, M. Y. Chen, and A. Hauptmann, "Comparison and combination of two novel commercial detection methods," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '04)*, vol. 2, pp. 1267–1270, Taipei, Taiwan, 2004.

[5] L. Y. Duan, J. Wang, Y. Zheng, J. S. Jin, H. Lu, and C. Xu, "Segmentation, categorization, and identification of commercial clips from TV streams using multimodal analysis," in *Proceedings of the 14th Annual ACM International Conference on Multimedia (MM '06)*, pp. 201–210, Santa Barbara, Calif, USA, October 2006.

[6] J. M. Gauch and A. Shivadas, "Finding and identifying unknown commercials using repeated video sequence detection," *Computer Vision and Image Understanding*, vol. 103, no. 1, pp. 80–88, 2006.

[7] X. S. Hua, L. Lu, and H. J. Zhang, "Robust learning-based TV commercial detection," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME '05)*, vol. 2005, pp. 149–152, Amsterdam, The Netherlands, July 2005.

[8] R. Lienhart, C. Kuhmuench, and W. Effelsberg, "On the detection and recognition of television commercials," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, pp. 509–516, Ontario, Canada, June 1997.

[9] N. Dimitrova, S. Jeannin, J. Nesvadba, T. McGee, L. Agnihotri, and G. Mekenkamp, "Real time commercial detection using MPEG features," in *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 1–6, Annecy, France, July, 2002.

[10] E. El-Khoury, C. Sénac, and P. Joly, "Unsupervised segmentation methods of TV contents," *International Journal of Digital Multimedia Broadcasting*, vol. 2010, 2010.

[11] J. Wang, L. Duan, Q. Liu, H. Lu, and J. S. Jin, "A multimodal scheme for program segmentation and representation in broadcast video streams," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 223–232, 2006.

[12] L. Liang, H. Lu, X. Xue, and Y. P. Tan, "Program segmentation for TV videos," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '05)*, pp. 1549–1552, Kobe, Japan, May 2005.

[13] Z. Zeng, S. Zhang, H. Zheng, and W. Yang, "Program segmentation in a television stream using acoustic cues," in *Proceedings of the International Conference on Audio, Language and Image Processing (ICALIP '08)*, pp. 748–752, Shangai, China, July 2008.

[14] S. Haidar, *Comparaison des document audiovisuels par matrice de similarité*, Ph.D. thesis, Paul Sabatier University of Toulouse 3, September 2005.

[15] X. Naturel, *Structuration automatique de flux vidéo de télévision*, Ph.D. thesis, University of Rennes1, Rennes, France, April 2007.

[16] S. A. Berrani, G. Manson, and P. Lechat, "A non-supervised approach for repeated sequence detection in TV broadcast streams," *Signal Processing: Image Communication*, vol. 23, no. 7, pp. 525–537, 2008.

[17] S. S. Cheung and T. P. Nguyen, "Mining arbitrary-length repeated patterns in television broadcast," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '05)*, vol. 3, pp. 181–185, Genes, Italy, September 2005.

[18] I. Döhring and R. Lienhart, "Mining TV broadcasts for recurring video sequences," in *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR '09)*, pp. 208–215, Santorin, Greece, July 2009.

[19] J. T. Foote and M. L. Cooper, "Media segmentation using self-similarity decomposition," in *Proceedings of the Conference on Storage and Retrieval for Media Databases*, pp. 167–175, Calif, USA, January 2003.

[20] C. Herley, "Argos: automatically extracting repeating objects from multimedia streams," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 115–129, 2006.

[21] K. M. Pua, J. M. Gauch, S. E. Gauch, and J. Z. Miadowicz, "Real time repeated video sequence identification," *Computer Vision and Image Understanding*, vol. 93, no. 3, pp. 310–327, 2004.

[22] J. Yuan, J. Meng, Y. Wu, and J. Luo, "Mining recurring events through forest growing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1597–1607, 2008.

[23] X. F. Yang, Q. Tian, and P. Xue, "Efficient short video repeat identification with application to news video structure analysis," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 600–609, 2007.

[24] M. Covell, S. Baluja, and M. Fink, "Advertisement detection and replacement using acoustic and visual repetition," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP '06)*, pp. 461–466, Victoria, Canada, October 2006.

[25] X. Wang, X. Li, Y. Qian, Y. Yang, and S. Lin, "Opportunities and challenges for next-generation applied intelligence," in *Studies in Computational Intelligence*, vol. 214, pp. 45–51, Springer, Berlin, Germany, chapter "Break-segment detection and recognition in broadcasting video/audio on C/S architecture", 2009.

[26] B. T. Truong, C. Dorai, and S. Venkatesh, "New enhancements to cut, fade, and dissolve detection processes in video segmentation," in *Proceedings of the 8th ACM International Conference on Multimedia*, pp. 219–227, Los Angeles, Calif, USA, November 2000.

[27] Orange, "Data mining software," http://www.ailab.si/ .